Carnegie Mellon

Software Architecture

Architectural Drivers

Instructor: Nguyễn Đức Mận
0904 235 945
mannd@duytan.edu.vn

Nội dung được xây dựng dựa trên tài liệu của Prof. Matthew Bass 2018 - ISR

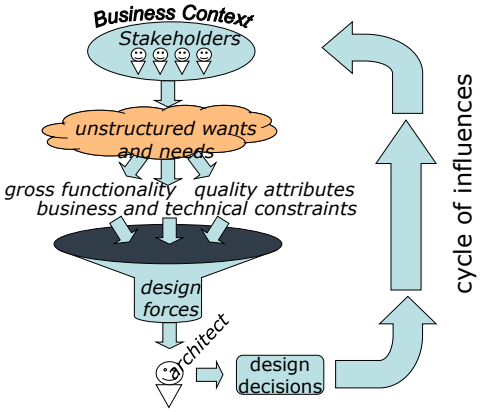
1

Session Outline

- What drives software architecture
  - Functional requirements
  - Constraints
  - Quality attribute requirements
- Examine typical expressions for quality attribute requirements
- A better way to express quality attribute requirements

2

Recall …



Typical Requirements Articulation

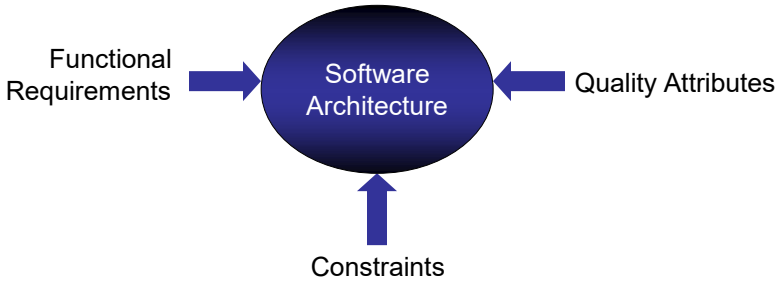
- Where do requirements normally come from?
- What do they look like?
- Do we differentiate architecturally significant requirements?

Breakdown

- Appropriately articulating the requirements are often first point of breakdown
- As an organization we need to understand and articulate the requirements that “drive” the software architecture

Architectural Drivers

- Architectural drivers are requirements that will shape the software architecture
- Architectural drivers include:



Functionality and Architecture?

What is the relationship between functionality and architecture?

Monolithic System Layered System

Functionally speaking which is better?

- Let's say these are options for the cell phone platform … functionally which one will make a "better" phone call?

Functionality and Architecture II

- Functionality is evaluated by comparing the output returned with what's expected
  - From this perspective these two options are equivalent
- The structure of the software doesn't impact the functionality that can be implemented
  - This means we can't use functionality as criteria for evaluating the "correctness" of a decomposition
  - What does this say about "traditional" OO and component based approaches?
- Functionality is important, however, and needs to be understood and accounted for
  - We will look in more detail at how this is done in the design section

Functional Requirements

- Functional requirements describe what the system must do.
- At architectural design time, we are concerned with high level function not details.
- As we create the architecture and refine it, we will learn more about the detailed functional requirements.
- For architectural design, functional requirements are best articulated as functional use cases.

Capturing Functional Requirements

- From an architectural perspective we are primarily interested in high level functionality
- Architects must engage stakeholders to describe how they will use or otherwise interact with the system
- There are many ways to capture them e.g.
  - Use cases
  - Feature descriptions
  - Operational descriptions
  - Formal specifications

Information to Capture

- With functional requirements we should consider:
  - Pre conditions (assumptions)
  - Primary “happy” path
  - Alternative flows
  - Error conditions
  - Post conditions (guarantees)

UC example

Use case	Description
Name	Payment of salary.
Description	It calculates the salaries for all the employees of the institution.
Actors	Head of accounting dept., accounting assistant.
Viewpoints	Accounting department.
Primary scenario	1. Get info about employees; 2. Calculate salaries; 3. Print cheque.
Includes	Cheque printing.
NFRs	Security, performance.

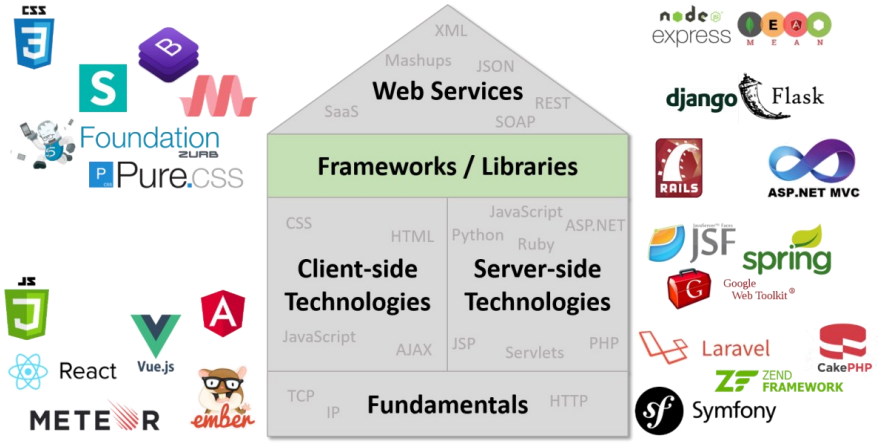
Constraints

- There are two kinds of constraints, technical and business
  - They impact the architecture differently
- Technical Constraints are pre-made design decisions.
  - some constraints are technical (direct)
  - other constraints are imposed by business concerns, the market place, and/or the organization (indirect)
- Sometimes we choose our own constraints.
  - if you need to store data, you will probably select a database
  - your selection will become a constraint in the system and will shape the architecture today and tomorrow

Technical Constraints

- Technical constraints are pre-made design decisions that become *load-bearing walls* on the design space.
  - Utilization of and/or integration with legacy systems
  - Mandated technologies, languages, protocols, standards, and so forth.
- While the origin of the constraint may differ, all will impact early architectural decisions to one degree or another.

Technical Constraints



Example Business Constraints

Business constraints don't always have obvious architectural design impact.

- Targeted market and time to market
- Cost and benefit expectation
- Rollout (incremental delivery) schedule
- Projected lifetime of the system
- Availability of workforce and allocation of expertise
- Alignment of team structure, available expertise, and software structures
- Product lines or other reuse strategy to amortize investments

Documenting Constraints

- We simply write constraints down including any supporting information that is relevant or available including:
  - rationale
  - any associated flexibility in the constraint
  - any suitable alternatives
  - originating stakeholder(s)
- There is no need to prioritize constraints according to importance – *by definition they are must be adhered to!*
  - It can be helpful to assign difficulty rankings

Quality Attributes, or "Non-Functional Requirements"?

- "Non-functional requirements" is a dysfunctional term
  - functional requirements usually describe the behavior of a system or the work it must do
  - non-functional requirements is a place holder for everything else like modifiability, performance, and so forth.
  - the term introduces a false partitioning of requirements – *try to describe a quality attribute without describing functionality*.
- The important thing is not "is it functional or not" but "is it architecturally significant"
  - In fact quality attribute concerns do have a functional aspect

Quality Attributes

- Quality Attributes – characteristics that the system must possess in addition to the functionality.
  - Of the architectural drivers, quality attributes are the most difficult to uncover, write down, and test.
  - Quality attributes and constraints drive architectural structure more than functionality.

Quality Attributes and Architecture

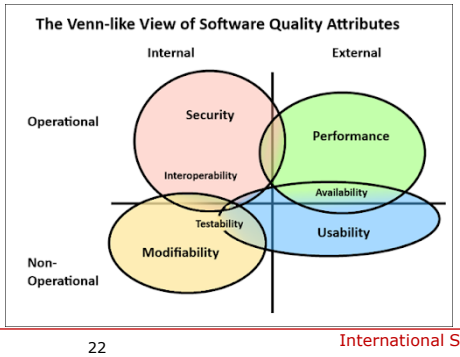
- Architectural choices implement functionality, meet constraints, promote some quality attributes, and inhibit others.
  - Architecture is critical to the realization of quality attributes.
  - A change in architecture that improves one quality may promote or inhibit the achievement of other quality attributes.
  - Architecture is critical to balancing *quality attribute tradeoffs* before detailed design, implementation, or investing in upgrades to a software intensive system.

Typical Descriptions System Requirements

- “The system response time for all actions shall be less than 0.5 sec”
- “Time critical functions like sending a command, switching pictures, graphics must have “sufficient” performance”
- “Fault tolerance shall be supported by the system -- such that the possibility and impact of server failure is minimized or eliminated. The availability and functionality of client workstations and substations must be maintained during failure conditions of servers or other clients.”
- “The system shall allow to shutdown and restart all RCM components in order to restart the complete system.”
- “The system shall have the capability detect software faults and automatically restart the corresponding component.”
- “flexible change and adaptation of HMI to create customer specific look and feel”

What Do You Think?

- Do these look familiar?
- What do they mean to you?
- How could they be improved?



Describing Quality Attributes – 1

- Quality attributes are often difficult to discover and describe.
- Names alone are ***not*** enough.
  - There is no widely accepted standard and utilized standard – *vocabulary varies widely.*
  - Descriptions are vague and lack quantifiable measure.
  - Simple textbook descriptions of quality attributes are usually too naive for use in practice.
  - Non-productive debates result from discussion of system “ilities” and what they really mean.

Describing Quality Attributes – 2

- For example, consider the following requirement: *“The system shall be modifiable.”*
- This is a meaningless requirement.
  - You cannot design a system that is modifiable for all potential changes.
  - Every system is modifiable with respect to some set of changes and not with respect to others.
  - What forbids system change is cost and schedule.
- The issue is not modifiability – the issues are:
  - How can we anticipate change early?
  - How can we plan and design for change?

Describing Quality Attributes - 3

- Quality attributes reflect some business concern
  - Modifiability is typically supports envisioned changes required by the business
  - Throughput is about supporting the anticipated load
- Quality attribute requirements are the criteria that we use to "test" the design
  - If we're not able to do this we won't know what we're going to get until we build it
  - If this is how we are going to proceed we will not be able to make informed business decisions
- How do we adequately describe these kinds of requirements?

Quality Attribute Scenarios

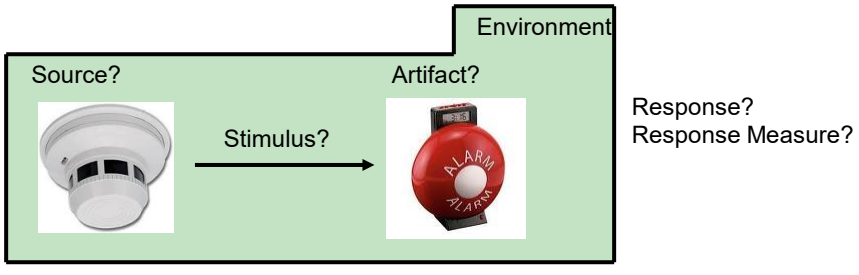
- *Quality Attribute Scenarios* are a means to better characterize quality attributes.
  - The first step is to quantify the quality attributes through scenarios.
  - Scenarios should be prioritized.
  - Then architects can make reasoned decisions regarding structures to balance all of the architectural drivers.
  - Once quantified, we can measure conformance, completion, and success.
- Conceptually Quality Attribute Scenarios are to quality attributes what Use Case Scenarios are to functionality

Six Parts of a Quality Attribute Scenario

1. **Stimulus** – A condition that affects the system.
2. **Source of the stimulus** – The entity that generated the stimulus.
3. **Environment** – The condition under which the stimulus occurred.
4. **Artifact stimulated** – The artifact that was stimulated by the stimulus.
5. **Response** – The activity that results because of the stimulus.
6. **Response measure** – The measure by which the system's response will be evaluated.

Six Parts of a Quality Attribute Scenario – 2

"A external smoke detector sends an alarm event under peak load. The building management system receives the event and raises an alarm within .5 seconds."



What quality attribute is being represented here?

Example Quality Attributes

- To illustrate the properties of various quality attributes, as examples we will examine
  - availability
  - modifiability
  - performance
  - security

Availability – 1

Definition: *Availability is concerned with system failure and its associated consequences. A system failure occurs when a system no longer delivers a service that is consistent with its specification.*

Availability – 2

- Areas of concern include
  - preventing catastrophic system failures
  - detecting system failures
  - recovering successfully from system failures
  - the amount of time needed to recover from system failures
  - the frequency of system failures
  - degraded modes of operation due to system failures

Availability – 3

Example Stimuli

- omission, crash, timing, events, response

Example Sources

- internal, external, software, hardware

Example Artifacts

- systems, processors, software, hardware, storage, networks

Example Environments

- operational, test, development, load, quiescence

Example Responses

- detect and notify, record, disable, be unavailable, continue operation in degraded mode

Example Response Measures

- critical time intervals when the system must be available
- availability time
- time intervals in which the system can operate in a degraded mode
- repair time

Availability – 4

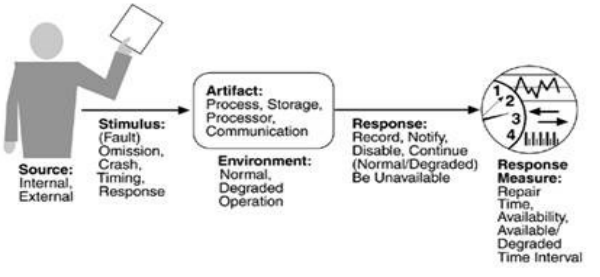
Example Availability Scenario:

"An unanticipated external hardware error is received by the initialization process during startup. The initialization process prevents an engine start and illuminates the check engine light."

5 mins for discussion

Modifiability – 1

Definition: *Modifiability is about the cost of change and refers to the ease with which a software system can accommodate changes.*



Modifiability – 2

- Areas of concern include
  - Identifying what can change.
    - functions, platforms, hardware, operating systems, middleware, systems it must operate with, protocols, and so forth
    - quality attributes: performance, reliability, future modifiability, and so forth
  - When will the change be made and who will make it.

Modifiability – 3

Example Stimuli

- add/delete/modify functionality or quality attribute

Example Sources

- end user, developer, system administrator

Example Artifacts

- systems, OS, hardware, software

Example Environments

- runtime, compile time, build time, design time

Example Responses

- locate places to be modified
- make modifications without side affects
- test the modification with minimal effort
- deploy the modification with minimal effort

Example Response Measures

- cost in terms affected, components, effort, and money
- extent to which this modification affects other functions and/or quality attributes

Modifiability – 4

Example Modifiability Scenario:

"A product manager request the system accommodate devices supporting the next version of the BACNET standard. The system can accommodate these devices within 1 person month of effort with no changes to the software architecture."

5 mins for discussion

Performance – 1

Definition: *Performance is about timing: how long it takes the system to respond when an event occurs.*

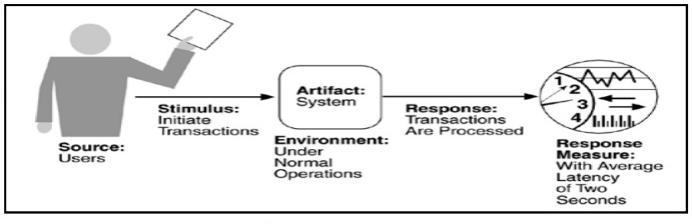


Figure 3: Sample performance scenario

Performance – 2

- Areas of concern include
  - varying sources of events: interrupts, messages, requests from users, transactions, and so forth
  - varying arrival rates and patterns: sporadic, periodic, stochastic, or some combination
  - response time: elapsed time from event arrival to system reaction

Performance – 3

Example Stimuli

•periodic, sporadic, or stochastic event arrival

Example Sources

•internal, external, software, hardware

Example Artifacts

•systems, OS, hardware, software

Example Environments

•operational, test, development, load, quiescence

Example Responses

•processes stimuli

Example Response Measures

•latency, deadline, throughput, jitter, miss rate, data loss

Performance – 4

Example Performance Scenario:

"500 users initiate 1,000 transactions per minute stochastically under normal operating conditions and each transaction is processed with an average latency of two seconds."

5 mins for discussion

Security – 1

- Definition: *the measure of the system's ability to resist unauthorized attempts to use data or services while providing access to legitimate users*

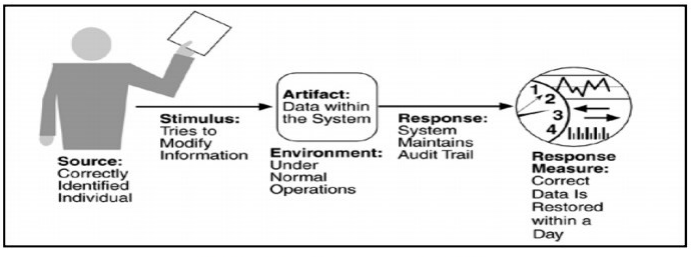


Figure 4: Sample security scenario

Security – 2

- Areas of concern include
  - non-repudiation: Transactions cannot be denied by any of the parties participating in the transaction.
  - confidentiality: Data and services are protected from unauthorized access.
  - integrity: System data and services are delivered as intended.

Security – 3

- Areas of concern include (continued):
  - assurance: The parties of a transaction are who they purport to be.
  - availability: The system will be available for legitimate use.
  - auditing: Activities are tracked within the system at levels sufficient enough for the reconstruction of events.

Security – 4

Example Stimuli

- entity tries to display data, change/delete data, access system services

Example Sources

- an unknown entity that is identified correctly/incorrectly, who is internal/external to the system, authorized/not authorized, and has access to limited/vast resources

Example Artifacts

- systems, services, data

Example Environments

- online/offline, connected/disconnected, protected/unprotected

Security – 5

Example Responses

- authenticate the user
- hide the identity of the user
- block access to data/services
- grant/deny access to data/services
- record access/attempts to access data, modifications to data
- store data in an encoded format
- recognize unexplainably high or unusual demands for services/data
- report/restrict access.

Example Response Measures

- time/effort/resources required to circumvent security measures with success, restore data/services
- probability of detecting attacks
- probability of identifying individual responsible for the attack
- percentage of services still available under a denial-of-services attack
- extent to which data/services were damaged and/or legitimate access was denied

Security – 6

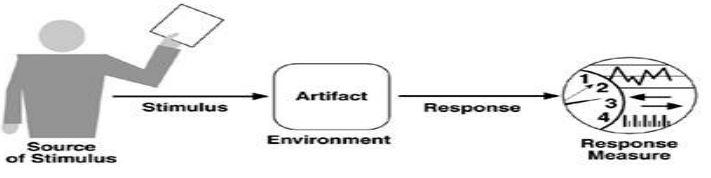
- Example Security Scenario:

  "A unauthorized individual from an external site gains access to the system and tries to modify consumer data. The system detects the malicious behavior, maintains an audit trail of the unauthorized individual's actions, notifies system administration, and shuts down the system."

5 mins for discussion in group

Classic Quality Attributes

- So far we have been discussing the following "classic" quality attributes as exemplars
  - availability
  - modifiability
  - performance
  - security
- The text [BCK03] lists others – this is not an exhaustive list of quality attributes.
- These are regarded as "first-class" quality attributes.



Quality Attribute Template

Scenario Title: Descriptive title if desired	Scenario ID: Pneumonic reference if desired
Raw Quality Attribute Description: This might be as simple as "modifiability" or an ill-formed requirements statement that tries to describe a quality attribute requirement or property of the system.	
Source of Stimulus:	This is a description of the originating source or sources or potential sources of the stimulus or stimuli.
Stimulus:	This is the phenomenon that prompts the system to react in some way. It might be a user request, an event or interrupt, an error, a request for change, and so fort.
Environmental Condition(s):	This is a description of any relevant environmental conditions that could affect the response and significant measures of the response. It might include such conditions as normal operation, peak load, degraded operation, at development time, and so forth.
System Element(s):	These are the system elements affected by the stimulus. Early development lifecycle (pre architectural design) the affected elements might not be knows. As design commences, scenarios should be refined and elements identified as they are known.
System Response:	This is the desired response of the system.
Significant Measures:	This is the significant measure by which the quality of the response will be judged. Response measures vary and may be in terms of person hours, error detection, response time, recovery time, and so forth.

Quality Attributes in Practice – 1

- Availability, Modifiability, Performance, and Security are fairly common quality attributes,... but there are others...
  - interoperability
  - safety
  - usability
  - extensibility
  - portability
  - scaleabilty
  - learnability
  - maintainability
  - testability
  - availability
  - buildability
  - and so many more...

Quality Attributes in Practice – 2

- In practice, there are other quality attributes that are not so familiar, for example: *calibrateability*
  - Some quality attributes are domain specific.
- Single word descriptions of quality attribute properties are meaningless – *empty vessels that any meaning can be poured into!*
- Its not important what term we use, but *make sure that it is characterized in a quality attribute scenario*.
  - This conveys the real meaning of the "ility."

Session Summary – 1

- Architectural drivers shape the architecture
  - includes high level functional requirements, technical and business constraints, and quality attributes
- Constraints and quality attributes will shape the architectural design of the system far more than functionality alone.

Session Summary – 2

- Quality attributes are difficult to discover and document.
- We discussed the use of quality attribute scenarios to characterize quality attribute scenarios.
- We discussed four popular quality attributes in detail and examine ways to characterize them through quality attribute scenarios.

References

- Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.
- Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; Wood, W. *Quality Attribute Workshops (QAWs), Third Edition.* Technical Report CMU/SEI-2003-TR-016: Software Engineering Institute, 2003.
- Lattanze, Anthony J. *Architecting Software Intensive Systems.* Boca Raton, FL: Auerbach, 2009
- Video summary: https://youtu.be/Xkvt06WgzTk